DSP Implementations for Accelerator Instrumentation

T. J. Shea
Brookhaven National Laboratory
Upton, NY 11973, USA

Abstract

Digital signal processing is now a fundamental component of many instrumentation systems. By presenting several example systems, this paper will review the functions provided by digital signal processing and discuss implementations on available hardware. The hardware platforms can be divided into three categories: reconfigurable computing systems based on field programmable gate arrays, programmable Digital Signal Processors (DSPs), and general purpose microprocessors. The provided functions include data rate reduction to decrease network load, local calculations to decrease computational load, and local real-time control. Although several examples will be presented, they all represent similar position monitoring applications that differ primarily in the signal processing implementation. Finally, the relative performance of the three implementations will be compared. Benchmark results and the ease of implementation will be discussed.

1 Introduction

The application of digital signal processing to accelerator instrumentation provides several benefits inluding network load reduction, system modularization, and real time feedback. Because modern measurement systems tend to digitize signals near the source, high data rates are produced that can overwhelm most control system networks. By including digital filtering and local buffering, the system designer maintains the stability and accuracy inherent in a digital front end, yet provides the control system with data at a useful rate. With signal processing capability built into the system, the designer is usually tempted to offload computations from the control console level and perform them locally. This practice can produce high performance, self contained instruments, but developing for this deeply embedded environment can be more challanging than programming at the console level. In systems that include beam in the loop, embedded digital processing is an absolute requirement rather than a nice feature. Because these systems typically have stringent requirements on throughput and latency, they allow little flexibility of implementation.

The designers of accelerator instrumentation apply signal processing technology in an environment that differs from most vendors' target markets. In an industry that signs contracts for 10,000 units per month, the small number of processors required by the following examples are not particularly interesting. Therefore, instrumentation designers must follow the technology to wherever the other markets drive it.

2 Functions provided by signal processing

Digital signal processing provides features that may be catagorized into three functional groups: data reduction, local calculation, and real-time feedback. The first two functions are most useful in the data acquisition applications that typify accelerator instrumentation.

2.1 Data reduction

A current trend in instrumentation design is to push digitization closer to signal source. Although measurement performance benefits, this technique generally leads to increased data rates that could overwhelm the control system network. Network load can be reduced by locally filtering or signal averaging the raw data stream. Additionally, the signal processing subsystem could provide circular data buffering to produce a virtual flight recorder. A subset of this data is then sent to the console level in real time while the entire buffer is delivered only after acquisition has stopped.

2.2 Local calculation

By performing calculations locally, the network load is not necessarily reduced, but some computational load is removed from the higher levels of control system. This can also lead to a convenient modularity whereby the system becomes a self-contained instrument. Early in the life of a system, during fabrication and installation, built-in self test can reduce the need for dedicated test equipment. During operations, this feature allows the system to produce its own calibration data and apply it to the data stream.

2.3 Real time feedback

In systems that require real time feedback, embedded signal processing is more than a convenience. These systems commonly include beam in the feedback loop. Therefore, deterministic latency is usually desired. Feedback can also be used in data acquisition

applications to control thermally induced drifts, or to automatically adjust gains and offsets.

3 Three implementations

For the purposes of this paper, all implementations will be catagorized into three types:

General purpose microprocessor - Popular examples include the Intel Pentium, and the Motorola/IBM PowerPC.

Programmable DSP - Floating point examples include the TI 'C40, and the Analog Devices SHARC. Fixed point examples include the Motorola 56k, and the AT&T DSP16x families.

Configurable hardware - These implementations usually employ in-system programmable gate arrays by Xilinx, Altera, and several others.

These three implementations offer a tradoff between performance and flexibility. The usual assumption is that the high performance gate arrays offer limited flexibility while the very flexible software-based implementations running on general purpose processors suffer from low performance. The programmable DSPs should fall somewhere in the middle. These assumptions seem to be born out in practice as the following examples will show. However, recent benchmarks reveal some surprises.

3.1 General purpose microprocessor

Most of these processors now have a complex architecture that includes deep pipelines and superscaler features. Because of this complexity, applications are best programmed in a higher level language. Hand optimization in assembly can be very challenging so the programmer usually relies on the compiler to produce executables that will avoid pipeline stalls and keep the multiple execution units busy. These features all lead to high average performance in complex calculations, but they also lead to unpredictable execution times. Because of this, cycle accurate simulators are rarely available. This nondeterministic behaviour also requires that the system designer leave plenty of processing headroom for time critical applications. Luckily, many instrumentation systems act as data acquisition systems with latencies bounded by human response times. These processors tend to have a wide variety of mature development tools available. Full-featured operating systems are also available that can ease development and debugging.

In the typical instrumentation system, a general pupose processor would usually be purchased as part of a board level solution running a real time kernal. Several systems use LabVIEW or similar software running on PCs that can be rack mounted or purchased as embedded VME, VXI, or CPCI boards. The CPU could also be shared with the control system front end.

As an example of this implementation type, consider a position monitor system at Jefferson Lab[1][2]. Figure 1 depicts a block diagram of the upgraded switched electrode electronics and its corresponding readout system. This application demonstrates the data reduction and local calculation functions that is provided by digital signal processing. Gain control feedback could be provided in the future. Four position monitors are serviced by each VMIC digitizer board. The digitizer board resides in VME and provides a continous, double buffered data stream. A commercial single board computer contains the processor that runs both the position monitor software and the EPICS IOC software. The position monitor software is interrupt driven at 15 Hz, coded entirely in C, and modularized into multiple VXWorks processes. It performs the usual gain and offset corrections, position calculations, signal averaging, and buffering. Hardware efficiency and integration are the hallmarks of this system.

3.2 Programmable DSP

In contrast to the complex processors discussed above, programmable digital signal processors rely on simple architectures that lead to deterministic execution times. Therefore, cycle-accurate simulators are available for nearly all DSPs. While the general-purpose processors are nice matches for current compiler technologies, typical DSPs have few compiler-friendly

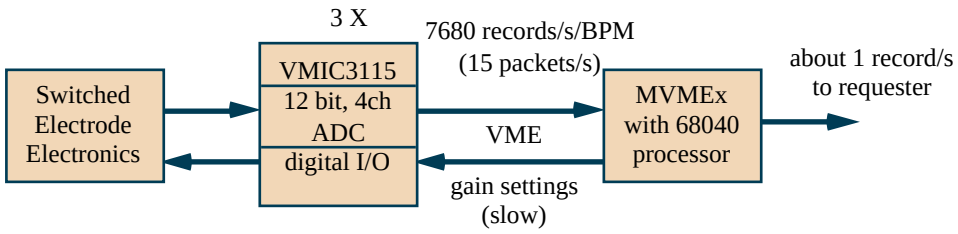

Figure 1. The upgraded position monitor system at Jefferson Lab.

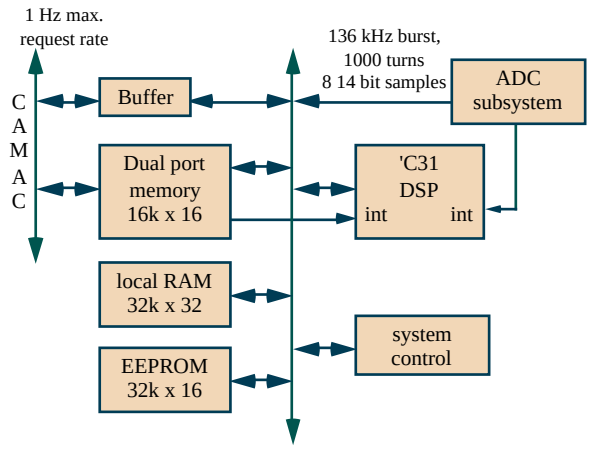

Figure 2. PEP II position monitor data acquisition diagram.

features. As an alternative, they allow straightforward assembly optimization of most signal processing algorithms. In fact, several features such as circular and bit reversed addressing are included exclusively to support digital filtering and FFT algorithms. Unfortunately, some of these features do not map well to the C language, so inline assembly or prebuilt libraries are used. Development environments for these processors tend to support simple applications, but most lack the tools that have become standard for developing large, modular applications.

Current instrumentation applications sometimes incorporate commercially available board level solutions. To reduce cost, systems requiring more than a few processors usually embed the DSP in system-specific circuitry. In these cases, the incremental cost is low and the DSP can sometimes reduce the cost further by replacing dedicated logic. In order of decreasing popularity, I/O interfaces include: hardwired, DSP-specific ports, IP/PMC, proprietary mezzanine, and system bus.

A specific example that takes advantage of DSPs is the SLAC PEP II Position Monitor Module[3][4] shown in Figure 2. Hundreds of these modules are distributed throughout the tunnel and each services four BPM channels. In contrast to the previous example, the processor runs no operating system and the software runs in a single thread utilizing polled interrupts. The real-time data buffering code is written in assembly, but all else is written in C. The functionality is similar to the Jefferson Lab system and includes data correction, position calculation, signal averaging, and calibration. Although the processor is deeply embedded, new code can be remotely downloaded into the onboard EEPROM. This implementation interfaces a modern instrumentation system to a mature control system. With its embedded DSP subsystem, each module functions as a self contained instrument.

3.3 Configurable hardware

If the general purpose processors have complex architecture, and the programmable DSPs have a simple architecture, then without significant exaggeration, configurable hardware could be described as having no architecture. This type of implementation normally uses fine grained in-system programmable gate arrays. Faced with a sea of gates, the designer has complete architectural freedom. In reality, most designers will make use of intellectual property supplied as synthesizable VHDL or vendor supplied macrocells. Recently, several vendors have begun offering signal processing libraries for their gate arrays. Current design tools still require significant expertise. Design entry methods include schematic capture, VHDL (or Verilog) coding followed by logic synthesis, and synchronous data flow diagramming. None of these tools approaches the convenience of the previously discussed

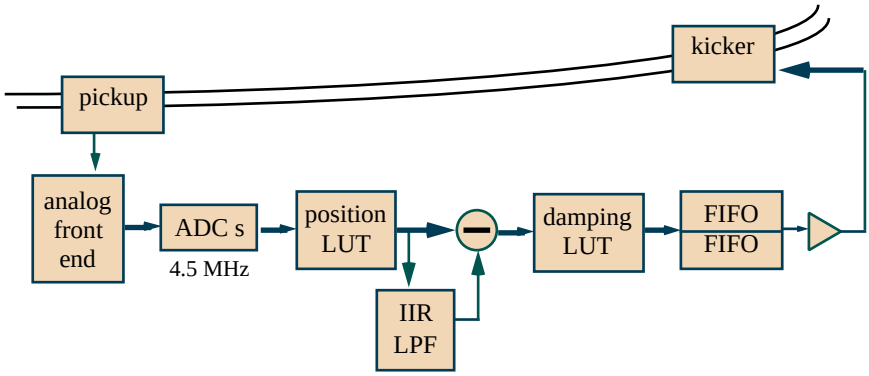

Figure 3. Brookhaven AGS transverse damper.

development systems, although some more advanced environments are now emerging from the universities.

Because standard interfaces can add significant complexity, configurable hardware is usually embedded within the application specific circuitry. Exceptions are beginning to appear in the form of commercial Industry Packs and PCI Mezzanine Cards that include user configurable gate arrays. Reconfiguration times are usually much less than a one second. This is fine for debugging, upgrading, and mode switching, but the time must be reduced even further to enable on the fly hardware reuse.

The AGS transverse damper[5] shown in Figure 3 is an instrumentation system that incorporates programmable logic in the signal processing chain. As in the previous examples, the signal begins in a position monitor electrode. Unlike those examples, this system also provides feedback and therefore must meet strict requirements on throughput and latency. Since orbit motion must be rejected at or below the synchrotron frequency, the output of the digital low pass filter is subtracted from the 4.5 MSa/s data stream. The filter is an IIR implementation that makes efficient use of a small logic array. Other functions, such as position calculation utilize lookup tables (LUTs) to maintain the required throughput.

4 Benchmarks

Since performance is such an important selection criterion for signal processing technology, this paper concludes with two benchmark summaries. The first is the ubiquitous FFT and the second is a simple set of operations found in most of the examples above.

A compilation of published benchmark results[6][7][8] is shown in Figure 4. The horizontal axis shows the time to compute a 256 point FFT normalized to the TI TMS320C54 fixed point DSP. Although the results were obtained from a several sources, every attempt was made to correctly normalize and obtain a valid comparison. All benchmarks were run at the device's native resolution and in its native format. As expected, the dedicated hardware shows the best performance. Surprisingly, the general purpose processors are now faster than most DSPs at this fundemental signal processing task. This is due to the recent improvements in the general-purpose processors' ALUs and to the clock speed advantage that these processors currently enjoy.

After seeing these results, the author organized some quick and dirty benchmarking at Brookhaven National Laboratory. The benchmark algorithm performs the following operations:

- read a value from zero wait state memory
- apply upper and lower limit checks
- apply third order correction
- accumulate for averaging
- check flag and branch

These operations are relevant for several systems at BNL and most are present in the previous examples. Figure 5 shows the maximum throughput obtained on several platforms. Again, the hardware solution exhibits the highest performance. Although the gate array cannot be clocked as fast as the other processors, this implementation makes use of distributed, pipelined logic to boost performance. Two programmable DSP systems were available for testing and they were outperformed by most of the general-purpose processors. The benchmark was also implemented in

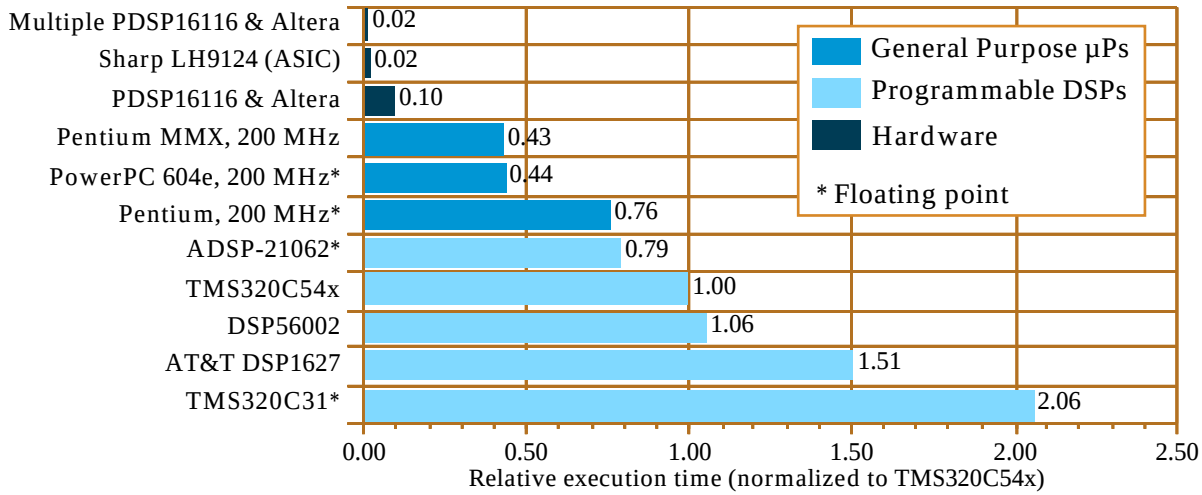

Figure 4. Execution time for a 256 point FFT.

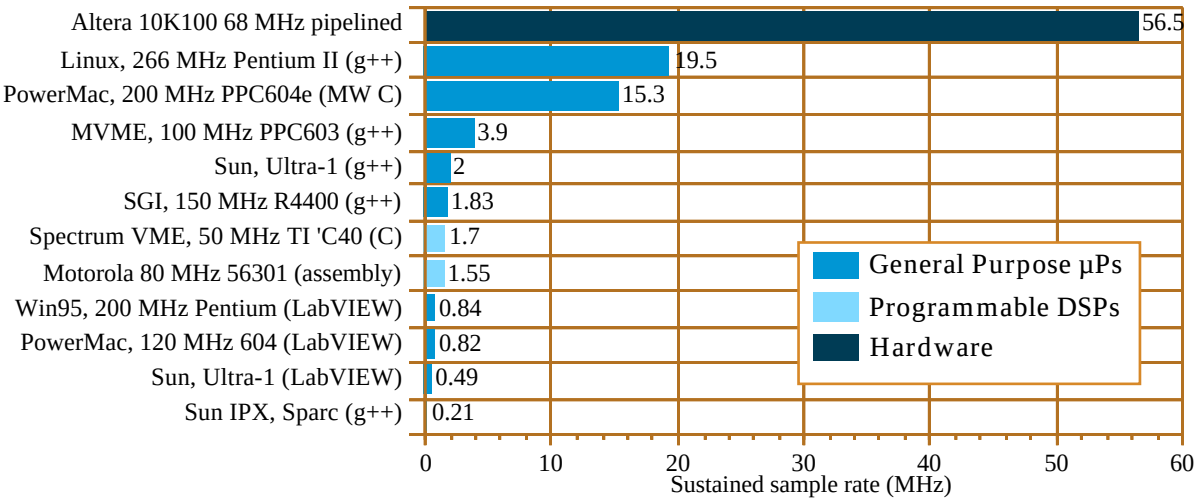

Figure 5. Results of a quick and dirty benchmark performed at BNL.

LabVIEW since the RHIC instrumentation group uses this environment to test prototype signal processing functions. Performance was near the bottom of the pack, but still respectable considering the high level development environment.

Of course, reporting the raw performance on small benchmarks does not tell the entire story. The ease of implementation for the latter benchmark was also noted. Development in C and LabVIEW took less than one hour. The C implementation ported easily although it had to be adapted to some platform specific libraries for timing. The resulting LabVIEW library ported flawlessly across all supported platforms. Development of the fixed-point assembly program took a bit longer to develop than the C version and the programmer did not have time to fully optimize it. The VHDL based gate array implementation took several hours to develop and was the most difficult to debug. The I/O performance was not explicitly tested, but a quick check on some platforms showed that performance was significantly degraded by VME access. The best system throughput and latency is probably still attained by embedding the processor within the application specific circuitry. As noted above, instrumentation systems constructed in this manner usually incorporate configurable hardware or programmable DSPs.

5 Acknowledgments

The quick and dirty benchmarking was performed by A. Campbell, C. Degen, L. Hoff, W. MacKay, J. Mead, and D. Shea.

The author is also grateful for enlightening discussions with the following individuals: E. Barsotti, C. Brayet, L. Hendrickson, M. Keesee, G. Vismara, T. Roser, W. A. Ryan, R. Santemaria, H. Schmickler, G. Smith, and S. Smith.

This work was supported by the U.S. Department of Energy.

6 References

[1] T. Powers, et. al., “Design, Commissioning, and Operational Results of Wide Dynamic Range BPM Switched Electrode Electronics”, Proc. of the 7th Accelerator Instrumentation Workshop, Argonne, IL, May 1996, AIP Conf. Proc. No. 390 (1997).

[2] M. M. Keesee, private communication.

[3] G. R. Aiello, et. al., “Beam Position Monitor System for PEP-II”, Proc. of the 7th Accelerator Instrumentation Workshop, Argonne, IL, May 1996, AIP Conf. Proc. No. 390 (1997).

[4] L. Hendrickson, private communication.

[5] G. A. Smith, et.al., “Transverse Beam Dampers for the Brookhaven AGS”, Proc. of the 5th Accelerator Instrumentation Workshop, Sante Fe, NM, October 1993, AIP Conf. Proc. No. 319 (1994).

[6] G. Blalock, “Microprocessors Outperform DSPs 2:1”, Microprocessor Report, Volume 10, Number 17 (1996).

[7] P. Lapsley, “DSP Benchmarks: Latest Findings”, The 1996 DSPx Exhibition and Symposium, San Jose, CA, March 1996.

[8] R. J. Peterson and B. L. Hutchings, “An Assessment of the Suitability of FPGA-Based Systems for use in Digital Signal Processing”, Proc. of the 5th International Workshop on Field-Programmable Logic and Applications, Oxford, England, August 1995.